大规模众核体系结构的并行模拟

叶笑春 范东睿 陈明宇 吕慧伟

摘要：随着芯片内部处理器核数的增多，多核处理器逐渐有向众核方向发展的趋势。而众核这一全新的体系结构给计算机模拟带来了挑战。串行模拟已经难以满足速度的需求，必须充分利用现有并行宿主机的多核资源，在保证不损失模拟精度的前提下提升模拟速度。本文以众核和众核集群两种体系结构为例，说明并行模拟技术在计算机并行体系结构模拟中的必要性和可行性，在众核模拟中，做到精度不变，模拟速度提升 10 倍；在众核集群模拟中，所模拟的处理器小核总数达到千核规模，并实现了混合的编程运行环境，为该结构的可扩展性测试提供了基础。

关键词：众核 众核集群 并行模拟

1 引言

在处理器体系结构的研究中，模拟器占有举足轻重的地位。模拟器技术贯穿于整个系统开发的过程之中：开发前期使用模拟器进行粗粒度模拟，以选择出最优的解决方案；开发期间对各种微结构进行验证；开发后期用模拟器进行软件开发及测试。在整个硬件系统投入运行后，还可以基于模拟器获取单纯依靠硬件系统无法得到的剖析（profiling）信息，对系统进行瓶颈分析及性能优化。在计算机基础研究过程中，模拟器可以模拟出各种现有体系结构甚至未来的新型体系结构，从而可以促进微结构和系统软件的研究。模拟器具有可控及可重复等优点，所以是洞察系统行为的重要工具。

衡量模拟器性能主要有两个指标：模拟速度和模拟精度。在传统的串行模拟中，这两个指标往往不可兼得，提高模拟速度通常会导致模拟精度的损失，反之亦然。所以，如何实现一个在保证模拟速度的同时，又能够保证模拟精度达到预期的模拟器是一个值得研究的问题。而在当前并行计算资源丰富的情况下，并行模拟显然是一个可行的方案。引入并行框架后，如何进一步模拟更大规模的众核体系结构，实现千核模拟的目标，也是本文关注的一个重要问题。

2 相关工作

2.1 Graphite

Graphite 模拟器是美国麻省理工学院在 2010 年发布的开源模拟器[1]，目标性能在 100 至 200 的减速比之间。它的主要特点是使用松散的同步机制，用牺牲模拟精度的方法来提高模拟速度。

2.1.1 Graphite 的执行方式

Graphite 是一个快速的，高层（high-level）的大规模多核模拟器。对于在模拟器上运行的应用程序来说，它对自己运行在什么样的宿主机上是完全不知情的。宿主机可以是一个

SMP[1]的多核机器，也可以是一个集群，而不需要对应用进行任何的修改。这与 SimK 并行模拟框架[2]很相似。在 SimK 中，单个宿主机和集群宿主机上运行同一个测试程序，并不需要对模拟的程序进行任何修改，只需要修改模拟器调用的 SimK 的应用程序接口（API）即可。

在 Graphite 中，应用程序的每个线程都被映射到所模拟的处理器核上，可以有一个或者多个线程被映射到进程中，多个进程可以运行在不同的宿主机上，也可以运行在同一个宿主机上。所以 Graphite 是一个两级并行的模拟架构。

另外，Graphite 使用动态的二进制翻译器产生指令，并对应用程序的数据访问进行了优化。它将应用程序的地址空间静态划分到不同宿主机上，并将与该地址空间相关的数据安置在相应的宿主机上，频繁访问的数据缓存到本地的内存上，这就解决了对在不同宿主机上数据的频繁访问可能形成瓶颈的问题。

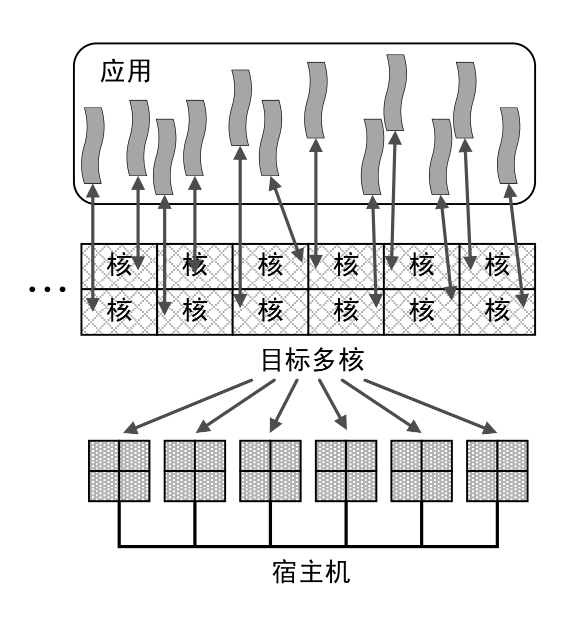


图1. Graphite 三层结构示意图

2.1.2 Graphite 的同步

Graphite 提供了三种不同的同步模型，用户可以根据自己的需要选择适合自己的同步方式。

Lax 同步

这种同步方式只在应用程序中发生同步事件时（包括栓锁（locks）、栅栏（barriers）、通过消息传递应用程序接口接收消息、线程产生和结束）才同步局部时钟，因此同步次数少，同步开销是最小的，模拟的减速比也最小。但这些事件可以乱序发生，所以这样做会使模拟的精度变差。

LaxP2P

这是一种很有特点的同步方式。人们经过观察发现，时间不精确的主要原因是由少数线程造成的，因此，可以每过若干个时钟周期（cycles）让每个目标核随机与另一个核组合成一对做比较，如果两者周期数相差太多，则周期数大的核暂停运行一段时间。使用这种同步方式能够获得较好的性能，同时也能获得较好的模拟精度。

LaxBar

这是一种类似于栅栏的同步方式。每过若干个周期，所有目标核进行一次同步。这可以保证核之间得到充分的同步，而且当同步周期间隔等于 1 时，整个模拟器可以达到时钟精确的模拟精度。但这种频繁的同步方式会直接导致性能的下降。

2.2 Cotson

[1] Symmetric Multi-Processing，对称式多重处理架构

COTson 是惠普公司基于 AMD 的 SimNow 模拟器开发的全系统模拟架构[3]。它可以模拟单核、多核甚至带互联网络的集群。模拟器的结构“可拆卸”，意味着用户可以用自己的设计来替换掉原有模块。

COTson 的一个设计原则是速度和精确度的权衡与折中，即可以牺牲模拟的精度来换取精度，或反之。这是时下较为流行的设计原则。随着众核的兴起，模拟规模随之增大，如何在可以容忍的时间内完成大规模模拟也成为一个研究热点，因此这种折中非常必要。

2.2.1 COTson 中的模拟技术

COTson 使用 SimNow 虚拟机做功能模拟，使用自己开发的时序后端来确定模拟目标的性能。COTson 的目标是实现大规模的全系统模拟，为了提高速度，采用了采样（Sampling）这种在快速模拟中经常使用的技术。采样的方法是将采样机制和模拟器耦合，采集模拟器中运行的指令和访存信息，再将这些信息传给时序后端，从而快速得到时序信息。

2.2.2 COTson 中的同步

在众核模拟中，目前采用较多的方法是并行离散事件模拟算法（PDES: parallel discrete event simulation）[4]，但是精确的并行离散事件模拟算法会在同步时产生较大的额外开销，拖慢模拟器运行速度。这在大规模模拟中是一个比较严重的问题，所以在一些不注重时序精确性的模拟器中就会对该算法进行改进，牺牲部分精度来换取更快的模拟速度。

COTson 中使用的同步方法是动态地调整同步粒度。顾名思义，“动态”的意思就是那些不关心的模拟部分就用较大粒度同步，而较为关心的模拟部分用较小的同步粒度，以换来更快的模拟速度。

从 COTson 中采样和同步这两个很重要的模拟技术来看，可以说 COTson 最主要的思想是找到我们关心的模拟点，放宽不重要部分的精度要求，达到精准和速度兼顾的目的。

2.3 其他

BGLsim[5]是 IBM 开发的 BlueGene/L 模拟器，运行在真实的 Linux 集群系统上，多节点通信使用 MPI[2]实现。BGLsim 模拟了大规模集群系统 BlueGene/L 的所有硬件，在这层虚拟硬件上运行 BlueGene/L Linux 系统，其上提供了移植的 BlueGene/L MPI 库。

MPI-SIM[6]是美国加州大学洛杉矶分校开发的一个 MPI 库，主要用来测试、调试以及预测并行程序在各种体系结构下的表现，它可以根据目标系统的参数特征给出不同的模拟结果，主要参数包括处理器个数以及通信延迟。MPI-SIM 是一个并行模拟器，它还提供了一种新的保守同步算法，减少了同步的开销和频率。

Simics[7]最初由瑞典计算机科学研究所开发，是一个全系统模拟器，可以模拟众多平台，并可无修改地运行多种操作系统。Simics 可模拟多种指令集架构，并拥有丰富的外设支持，在并行方面，Simics 也提供了基于 Link 机制的并行模拟方法。

3 众核模拟

上述提到的一些模拟器都有着各自的特点，而我们在众核模拟方面也进行了自己的尝试。我们前期在并行框架以及众核模拟方面有着一定的积累，开发了串行的众核 Godson-T

[2] Message Passing Interface，一种基于消息传递的并行程序设计标准

模拟器（GAS）[8,9]以及 SimK 并行模拟框架[2]。为了进一步加速众核模拟的速度，我们借助于 SimK 将 GAS 模拟器直接并行化，得到一个在模拟速度和模拟精度两个方面都能兼顾的并行众核模拟器：P-GAS。

3.1 SimK 并行模拟框架

SimK 是一款为了支持高效并行模拟[2]，并最大限度地支持通用性、可用性而设计的开源并行模拟框架。它满足超大规模支持、高可扩展性、异构宿主机支持、接口通用性和简洁性等方面的支持。

SimK 使用并行离散事件模拟算法同步机制。该算法的同步机制的核心思想是如果系统中任何局部都是同步的，则整个系统就是同步的。SimK 采用的是保守并行离散事件模拟算法同步方案。即局部因果关系需要严格保证，任何可能违背因果关系的操作都被禁止，在确保它的事件列表中最小时间戳事件可以被安全处理之前，该逻辑单元被阻塞执行。

另外，为了使模拟框架的效率更高，SimK 对 Pthread 进行了高度的优化，采用了无锁的同步机制，并支持在用户级对线程进行调度。

SimK 向外提供了应用程序接口（API），可以方便地将模块化的模拟器并行化，在本章主要介绍如何使用 SimK 模拟框架对 Godson-T 众核模拟器 GAS 进行并行化。

3.2 GAS 模拟器

Godson-T 模拟器（GAS）是一款串行模拟 Godson-T 芯片的事件驱动模拟器。Godson-T 芯片是一款高性能的众核芯片，片内含有 64 个处理器核，采用 2D-Mesh 的片上网络结构，处理器核之间及与 L2 缓存的通信通过片上的路由（router）进行，其结构如图 2 所示。模拟器 GAS 的目标是为 Godson-T 开发提供一个模块化的、可配置的模拟工具。它使用事件驱动的方式来模拟目标系统，即模拟模型由事件驱动，从而从一个状态转变成另一个状态。在众核系统中，事件会在不同的模拟组件中产生，模拟组件包括处理器核、路由、二级缓存等。GAS 使用全局队列来控制事件的处理，并且是时序精确的。

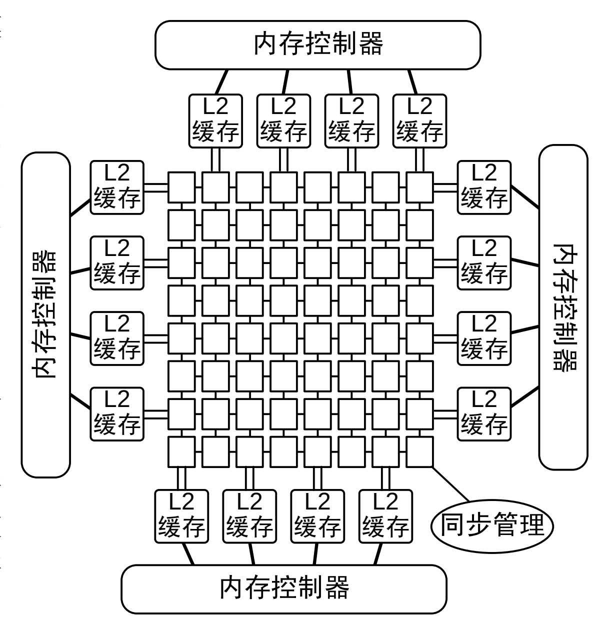


图2. 拥有 64 个处理器核的 Godson-T 模拟器结构

P-GAS 模拟器使用 SimK 模拟框架将 Godson-T 众核模拟器 GAS 并行化，并行目标是：将 Godson-T 模拟器中的模块划分到不同的线程上，并行运行模拟任务，取得较好的加速比，并保证结果正确及模拟精度不受损耗。在并行的过程中，为了得到更好的可扩展性，要尽可能减少模拟器原本代码和并行模拟器框架的代码修改量。

3.3 P-GAS 的并行化流程

模块的划分

为了提高 P-GAS 模拟器的运行效率，在模块划分时必须考虑组件之间的关联性、通信量、负载平衡等因素，任何一个因素处理不当都可能导致并行后的系统有负载不均或通信量大、同步开销高等问题。

封装模块

在模块划分完成后，还需要对每个划分出的子系统加入一个独立的事件队列，同时加入 SimK 同步通信应用程序接口，将其封装成一个 SimK 式的模块，封装后的模块如图 3 所示。

SimK 中模块间通信都是通过一个 SimK 通信通道来进行。模块产生消息后，将消息放入对应通道的缓冲区，待到通道所连接的另一个模块被调度时，会查看自己所有的通信通道，将所有收到的消息取出。

在串行版本的 GAS 模拟器中，所有的模块共享同一个全局队列，由于队列随模拟器运行会变得很长，造成事件插入队列开销过大的问题。P-GAS 模拟器中模块划分完成后，为每一个模块添加一个局部队列，取消原来的全局队列，以模块为单位管理队列。因为大部分模块的负载相对均衡，队列长度和使用的内存空间都可以方便地配置管理。

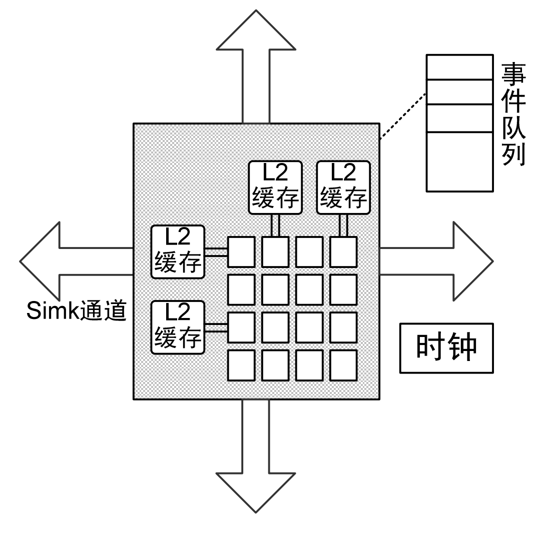


图3. 封装后的模块示意图

改进 SimK 的同步通信机制，使其适应 GAS 模拟器的特殊性

P-GAS 模拟器沿用 SimK 的保守提前量（lookahead）同步，这种机制能正确运行的前提就是模块不会收到其周期值小于模块当前自身周期值的事件。GAS 模拟器中存在一种 0 延迟事件，主要用于模拟信号线发送 `REQ` / `ACK` 或者 `REQ` / `NACK` 的情况。这种事件的出现会导致发送 `REQ` 的模块在时钟改变之后才收到其他模块返回 `ACK/NACK`，而此时的模块时间大于所需处理事件的时间戳，直接导致 SimK 的保守 lookahead 同步机制在并行 GAS 模拟器的工作中产生运行时错误。

解决的方法是引入一个 0 延迟事件的计数器，给每个模块增加一个标识变量 `flag`，其初始值为 0，当本模块的处理函数有 `REQ` 请求要发送到其他模块的时候，就将 `flag` 增加 1，每次收到一个 `ACK` 或 `NACK` 时，就将 `flag` 减少 1。根据这个 `flag` 标志，模块可以很简单地判断出当前周期是否仍会有自己要求其他模块发送的事件到来，从而根据这一点来决定时钟是否可以前进。

4 众核集群模拟

在实现了将并行离散事件模拟算法运用到单芯片内众核模拟器的并行加速后，我们希望将处理器核的数量增加，以验证该方法在芯片规模扩大之后的可扩展性，并希望针对千核规模的众核芯片做一些探索性研究。针对现有模拟器，我们将千核模拟作为一个目标。直观来看，针对我们希望研究千核体系结构的需求，有两种解决方案：第一，将原有的 GAS 模拟器扩展为拥有 1024 个处理器核的串行模拟器，然后采用 SimK 将其并行，完成单片千核的模拟；第二，将多个 GAS 模拟器以集群的方式进行互连，组成一个众核的集群模拟器，该

集群也可采用 SimK 作为并行框架进行并行加速。

经过对比分析，我们采用了第二种方案。原因在于第一种方案中，修改现有串行模拟器的工作量较大，且扩展性不佳。而采用集群的方式进行模拟，对现有的串行模拟器只需添加一些模块，并且也能使模拟器方便地在多个宿主机上工作，可扩展性可以得到良好的保证。

我们的目标是开发一个时钟精确的众核集群模拟器，基于 GAS 模拟器和 SimK 两个工作的积累，对 Godson-T 集群进行模拟，并在该集群上移植通信库，运行简单的应用程序。我们所希望模拟的系统结构如图 4 所示。为此，我们开发了一个用于模拟该目标结构的众核集群模拟器——P-Gcluster。

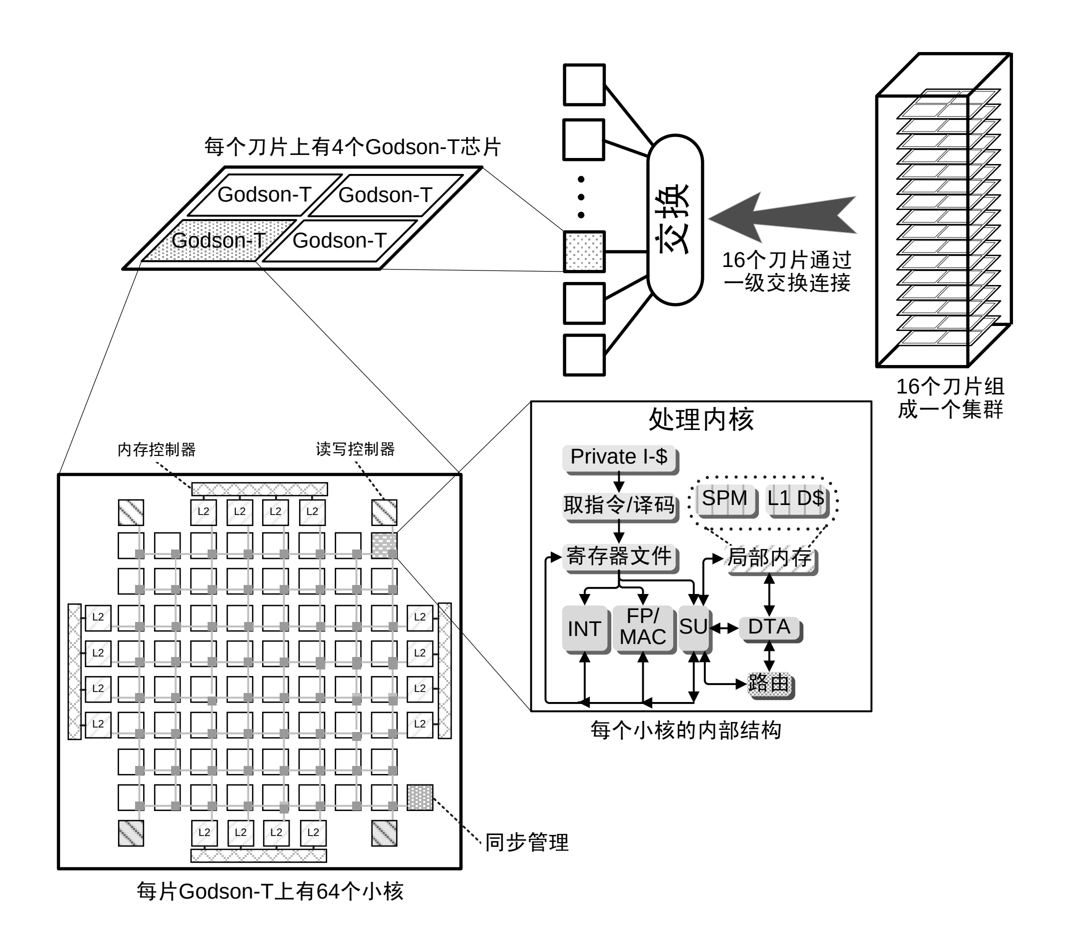


图4. 拥有 4096 处理器核的 Godson-T 集群结构

4.1 P-Gcluster 中网卡及交换机模块的设计

在原有的 GAS 模拟器中，没有与外界的交互模块，所以如果将多个 GAS 模拟器通过互连的方式组成千核规模的集群方式，需要为 GAS 模拟器增加一个网卡模块，如图 5 所示，主要模拟网卡的控制寄存器、状态寄存器和数据缓冲区。

而使用交换机将各个 GAS 模拟器的网卡进行连接，就可以组成一个众核集群。其中，网卡作为 GAS 模拟器与交换机之间传输数据的桥梁。在 GAS 模拟器中，网卡发送一个数据包的过程可以概括为三步：第一，GAS 模拟器往网卡缓冲区写数据；第二，GAS 模拟器给网卡发一个事件；第三，网卡往交换机发送数据包。详细流程如图 6 所示。

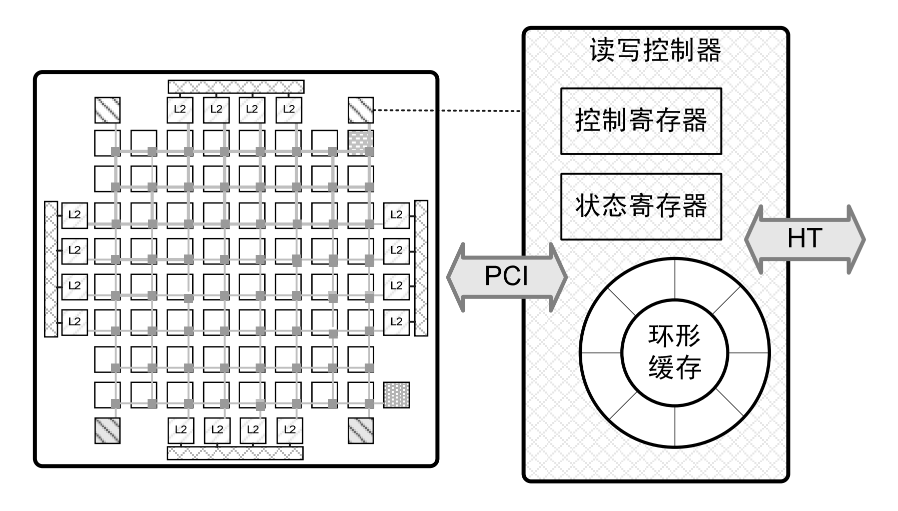


图5. GAS 上网卡结构示意图

对于网卡的模拟，不同的精度体现在网卡往外发包这一步，但是网卡模块与处理器核的接口是一样的。在网卡设计中，我们根据模拟的粒度不同，设计了简单网卡和基本网卡两种。这两种网卡均使用数学建模方式模拟各个阶段延迟，而基本网卡相对于简单网卡增加了数据包在缓冲区中的排队延迟以及流量控制机制。

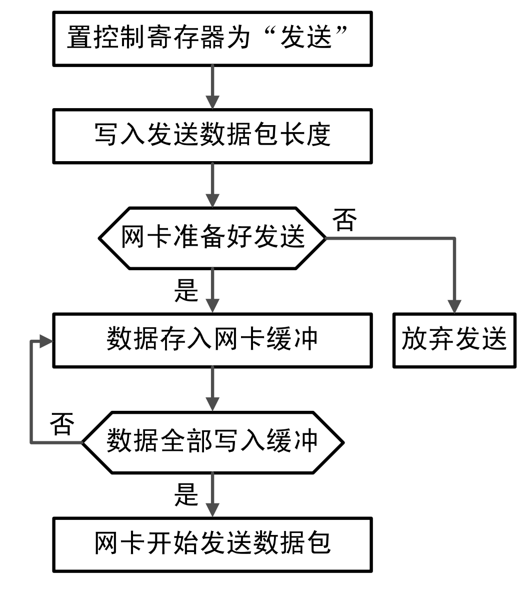


图6. 网卡发送数据流程图

为了真实模拟硬件环境中网卡的工作过程，在我们的基本网卡中，加入了流量控制机制。流量控制是网络中各个组件之间用于发送和接收流量控制单元的一种同步协议，由它决定数据包的发送时机。流量控制的目的是在不引起缓冲区溢出的前提下确保数据可以成功地由数据发送方传递到数据接收方，也就是说流量控制是交换收发双方缓冲区状态的一种方式。

在我们的系统中，我们采用了基于相对信用（credit）的流量控制机制[10]。首先设置信用为接收缓冲区的数量，它也代表接收方接收数据包的能力，该机制的工作原理如图 7 所示。

在模拟器中，网卡模块作为 GAS 模拟器的补充，负责 GAS 模拟器与外界的通信。而多个 GAS 模拟器之间则需要交换机模块进行连接。在系统中，交换机模块作为 SimK 的一个组件，在 SimK 中注册，并由 SimK 进行调度。

交换机功能可以分为数据转发和控制两个部分，其中数据转发功能负责数据的快速传递，由交换结构和排队结构组成；而控制功能负责交换机的配置管理以及路由填充等。控制功能一般由软件实现，此处不予讨论。

按照单个 GAS 模拟器内拥有 64 个处理器核的规模，我们在集群模拟器中，需要 16 个 GAS 模拟器才能达到 1024 个处理器核的规模。所以我们在设计交换机模块时，为其设置了 16 个端口，分别连接 16 个 GAS 模拟器的网卡模块。

当 SimK 对交换机运行进行调度时，交换机轮询自己的 16 个端口，检查是否有数据包

到达。有数据包到达时，接收数据包，将其放到交换机的缓冲队列中，并重复检查端口是否还有没有接收的数据包，直到所有数据包接收完成。然后识别接收到的数据包上所带的时间戳，若与当前时间一致，则本周期将数据包发送出去，否则交换机此次的调用结束，直到下一次被SimK调用。

上面已经提到，为了避免数据包在传输过程中的丢失，更加精确地模拟真实硬件，我们在基本网卡中添加了流量控制机制。为配合基本网卡中的流控机制，交换机模块也增加了对流控的支持。为了模拟数据包在交换机中的排队延时，我们设置了 1 个时钟周期内交换机可以发送数据包的个数上限。在 1 个时钟周期内，即使还有数据包没有处理完毕，交换机也必须停止这一时钟周期内的工作，并将所有未处理的数据包上的时间戳加 1，以便下一个时钟周期继续发送。

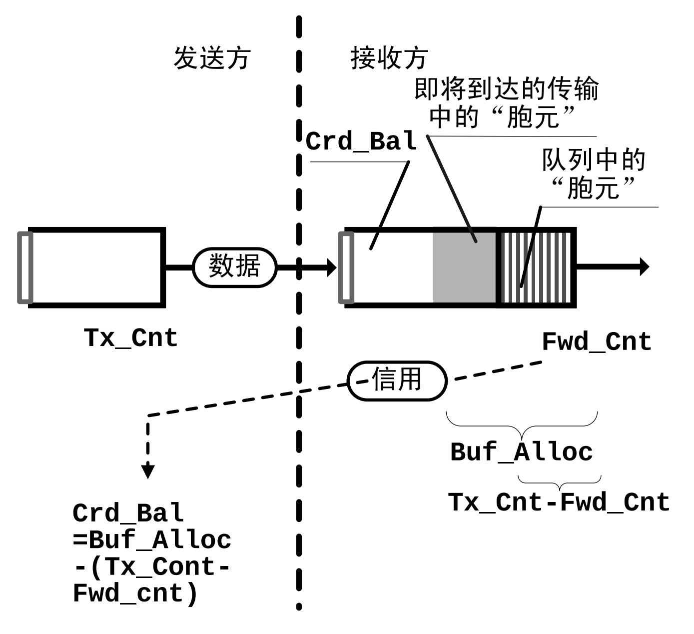


图7. 基于信用的流控机制原理

4.2 P-Gcluster 中片间互连网络的设计

在多个 GAS 模拟器进行互连时，要在模拟的第一步实现无阻塞的网络，只给出数学分析的传送包的延迟，不模拟包对缓冲区的竞争情况。众核集群的多个节点可以有多种方式进行互连。最为简单、成本最低的一种结构是总线式互连。这种结构在无竞争的情况下，传输延迟还可以保持在比较低的水平，但是随着节点数量的增多，总线上的竞争会越发明显，而此时总线也会成为节点间通信的瓶颈，不利于节点的扩展。针对集群系统，一般有以下几种互连可供选择：交叉开关互连、二维 Mesh（网状网）互连或者胖树互连。

综合几种互连的特征以及我们目前的模拟规模，我们选择了胖树互连的方式。胖树在很多高性能计算系统中得到了广泛的应用。由于胖树的主要特点是等分带宽，克服了一般二叉树或者多叉树中可能出现的根节点成为系统瓶颈、没有冗余通路以及容错性能交叉的缺点，从叶节点到根节点的通信带宽逐步增加，包含众多冗余链路，为实现可靠性提供了基础。且胖树能够满足高带宽、低延迟和可扩展的需要。

P-Gcluster 的整体拓扑如图 4 所示。每个刀片由 4 个 GAS 模拟器组成，由 16 个刀片组成一个小的集群。在集群中，可以采用比较典型的 m-port n-trees（m-端口 n-树）结构，在我们的模拟中，集群结构采用的是一级胖树 16-port 1-tree 结构。

4.3 P-Gcluster 上的两级并行环境

拥有了网卡和互连网络之后，集群模拟器的平台部分基本成型，但在这个平台上运行并测试程序还需要有通信库的支持，以便在平台上进行开发和运行程序的工作。在 GAS 内部，经过 SimK 的并行化，已经支持 pthread 的多线程运行。但是，在集群模拟器中，为了达到可扩展性的目标，我们在集群模拟器上移植了 MPI[11]，以便在此平台上进行应用程序的开发。

首先，为了测试 MPI 中六个基本的函数，我们在众核集群模拟平台上移植了简化版本的 MPI——gMPI，通过这个简化 MPI 的测试，验证了 MPI 在该平台上可以正确运行。接着我们又移植了目前被广泛采用的 MPICH，以更好地支持 MPI 的所有函数，并方便应用程序在集群模拟器上的开发。

通过 MPICH 的移植，我们搭建了一个两级并行的混合编程环境：在多个处理器芯片之间通过消息传递进行片间通信，而在单个处理器芯片中，各个处理器核通过 pthread 并行运行，与众核集群的结构比较吻合。

4.4 P-Gcluster 多进程多宿主机的扩展

由于宿主机资源有限，随着所模拟系统规模的增长，如果需要模拟的节点数超过宿主机上拥有的物理处理器核数量，在单个宿主机上进行模拟时宿主机性能会成为模拟的瓶颈。为了进一步提高模拟性能，我们需要将宿主机进行扩展，在宿主机集群上模拟目标集群系统。

在跨主机的模拟中，需要解决如下问题：

1. 底层网络通信需要通过套接字（Socket），需要 MPI 的支持；

2. 在多宿主机上运行模拟器时，需要一个统一的运行时环境，支持单一系统映像，负责给每个宿主机进行编号，任务分配并支持并行编程中的集合操作等。

5　实验

模拟器最为重要的两个评价指标为模拟器的速度以及模拟器的精度。我们在本文所描述的众核模拟器平台和众核模拟集群上分别对其进行了测试，以下是一些测试结果。

对于众核模拟器 P-GAS，我们采用 SPLASH-2 Kernel[12]并行程序测试集，它是一个被广泛使用的用于测试分布式共享存储多核处理器的测试集。测试程序包括了很多使用率非常高的科学计算应用，所以非常适合用于测试高性能计算系统。本文使用的实验平台为 4 CPU 的 AMD Opereton 8347SMP 系统（总共 16 核）。图 8 为在 P-GAS 模拟器上运行 SPLASH-2 Kernel 所得到的加速效果。

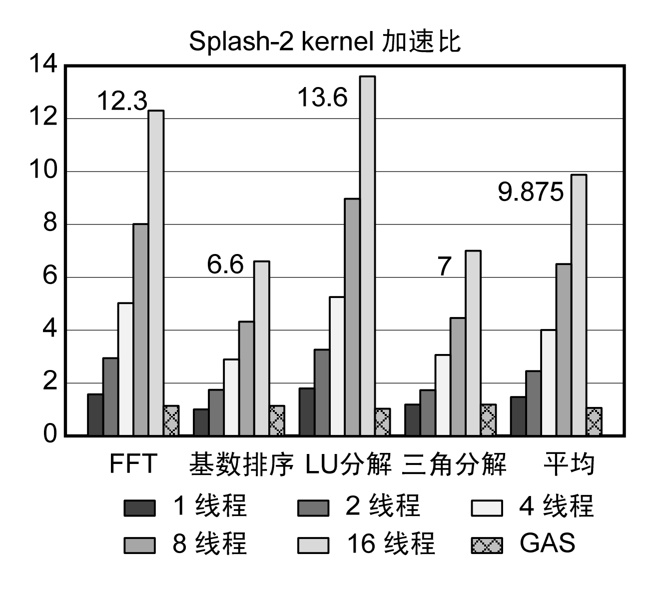


图8. P-GAS 模拟器上多线程并行运行 SPLASH-2 的加速效果

图中 FFT：快速傅里叶变换

表 1 列出了 P-GAS 模拟器在不同线程运行 SPLASH-2 时的周期数与原串行模拟器 GAS 相比的差异。可以看到，P-GAS 模拟器在 16 线程运行时保证性能得到平均 10 倍提升的同时，也保持了很高的精度。实现了一个在速度与精度两方面都能达到要求的并行众核模拟器。

对于众核集群模拟，我们主要测试了 P-Gcluster 的可扩展性。我们在众核集群模拟器上测试了 cpi、点积乘以及矩阵乘法等测试用例。得到的性能效果如图 9 所示。

表1. P-GAS 上多线程并行运行 SPLASH-2 与串行版本的周期数误差

	FFT*	基数排序*	LU 分解	三角分解	全部
16 线程	0.01%	0.07%	0.01%	0.17%	0.07%
8 线程	0.00%	0.44%	0.01%	0.14%	0.05%
4 线程	0.01%	0.07%	0.00%	0.02%	0.01%
2 线程	0.01%	0.51%	0.00%	0.02%	0.01%
单线程	0.01%	0.01%	0.00%	0.14%	0.06%

*快速傅里叶变换，*Radix

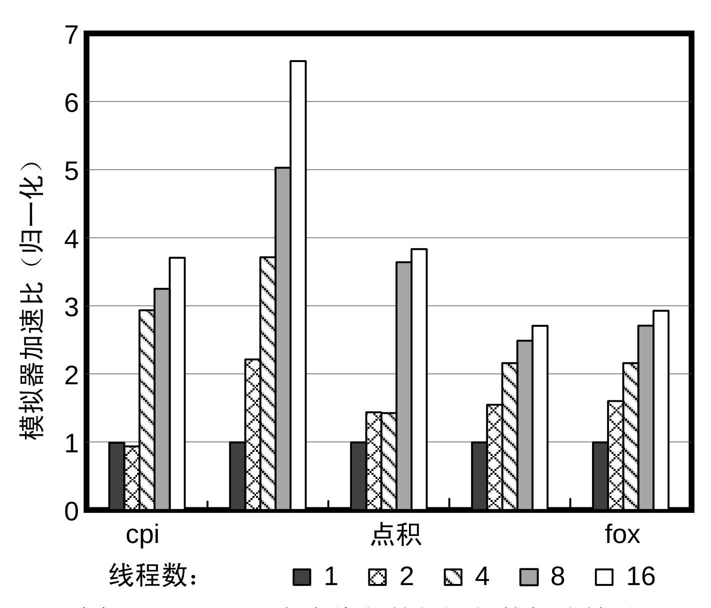


图9. P-Gcluster 上多线程并行运行的加速效果

图中 cpi：计算π值的算法；fox：一种矩阵乘法的并行算法。

我们对多进程的情况也进行了一些测试，主要测试了点积乘和矩阵乘法在多进程下的可扩展性，其结果如图 10 所示

可以看到，随着进程数量的增加，众核集群模拟器中各进程间需要传递的数量增多，网卡以及互连网络仍有进一步的优化空间。

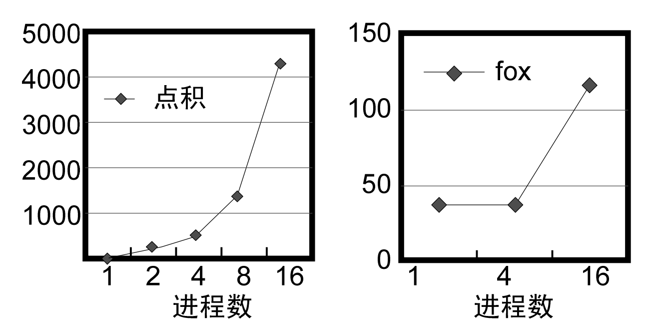


图10. P-Gcluster 上多进程并行运行的结果

6　总结

众核已经成为当今芯片发展的趋势，而模拟器在芯片开发中起了关键的作用。如何提高众核模拟器的模拟效率，使其模拟规模扩大就成为一个研究热点。P-GAS 是 Godson-T 体系结构模拟器的并行版本。为了能够使 P-GAS 的模拟效率更高，我们拆分了 GAS 模拟器的全局队列，根据 GAS 模拟器的拓扑进行了有效的模块划分。在高效 SimK 并行框架基础上，我们有效地解决了 P-GAS 模拟器的零延迟同步问题。

我们从加速比和精度损耗两个方面对并行效果进行了分析。模拟器达到了平均加速 10 倍的目标，最高加速达到 13.6 倍；精度方面，损耗最高只有 0.5%，平均不超过 0.1%，对模拟器的体系结构的研究不会造成大的影响。

进入 2010 年后，“如何模拟千核”成为模拟器界的热门话题。GAS 模拟器的模拟目标虽然为众核，但是 64 核的数量离千核仍然存在一定差距，所以我们以 P-GAS 模拟器的工作为基础，开发了一个众核集群模拟器，同样使用 SimK 作为并行模拟的框架。随着模拟处理器核数量的增加，模拟集群可以部署在宿主机集群上，实现多核集群模拟众核集群，充分发挥现有多核集群平台的优势，快速精确完成千核并行模拟。

参考文献：

[1] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald III, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, “Graphite: A distributed parallel simulator for multicores,” in HPCA ’10: The 16th IEEE International Symposium on High-Performance Computer Architecture, 2010.

[2] J. Xu, M. Chen, G. Zheng, Z. Cao, H. Lv, and N. Sun, “Simk: a parallel simulation engine towards shared-memory multiprocessor,” *Journal of Computer Science and Technology*, vol. 24, no. 6, pp. 1048–1060, 2009.

[3] E. Argollo, A. Falc´on, P. Faraboschi, M. Monchiero, and D. Ortega, “Cotson: infrastructure for full system simulation,” *SIGOPS Oper. Syst. Rev*., vol. 43, no. 1, pp. 52–61, 2009.

[4] R. M. Fujimoto, “Parallel Discrete Event Simulation”. *Commun, ACM*, 1990. 33: 30-53.

[5] Ceze, L., et al.Full Circle: Simulating Linux Clusters on Linux Clusters. Proceedings of the *Fourth LCI International Conference on Linux Clusters:The HPC Revolution 2003*, 2003

[6] Bagrodia, S.P.a.R. MPI-SIM: Using Parallel Simulation to Evaluate MPI Programs. *Winter Simulation Conference*, 1998: p. 467-474

[7] Magnusson, P.S.C., M.; Eskilson, J.; Forsgren, D. Simics: A full system simulation platform. *Computer IEEE*, 2002. vol.35, no.2: p. 50-58

[8] Dongrui Fan, Nan Yuan, Junchao Zhang, et al. Godson-T: An Efficient Many-Core Architecture for Parallel Program Executions. *Journal of Computer Science and Technology (JCST)*, 2009, vol.24, no.6, pp.1061-1073.

[9] Dongrui Fan, Hao Zhang, Da Wang, Xiaochun Ye, Fenglong Song, Junchao Zhang, and Lingjun Fan. High-Efficient Architecture of Godson-T Many-Core Processor, In Proceedings of *23rd Symposium on Hot Chips*, August 2011

[10] H.T. Kung, T. Blackwell, A. Chapman, “Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing”. Proceedings of the *ACM SIGCOMM 1994 Symposium on Communications Architectures, Protocols, and Applications*, Aug. 1994: 101-114.

[11] 都志辉，“*高性能计算之并行编程技术——MPI 并行程序设计*”，清华大学出版社，2001.

[12] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. “The SPLASH-2 Programs: Characterization and Methodological Considerations”. Proceedings of t*he 22nd International Symposium on Computer Architecture*, pages 24-36, Santa Margherita Ligure, Italy, June 1995.

作者简介：
叶笑春： 中国科学院计算技术研究所助理研究员 yexiaochun@ict.ac.cn
范东睿： 中国科学院计算技术研究所副研究员、博士生导师 Fandr@ict.ac.cn
陈明宇： 中国科学院计算技术研究所研究员、博士生导师 cmy@ict.ac.cn
吕慧伟： 中国科学院计算技术研究所博士生